Computing and Informatics, Vol. 32, 2013, 1001–1015, V 2014-Apr-9

TOWARDS FEDERATED CLOUD IMAGE MANAGEMENT

Álvaro Simón, Esteban Freire, Iván Díaz, Alejandro Feijóo, Pablo Rey, Javier López, Roberto Rosende, Carlos Fernández

Centro de Supercomputación de Galicia
CESGA
Av. Vigo S/N
15705 Santiago de Compostela, Spain
e-mail: grid-admin@cesga.es

Owen Synge

SUSE LINUX Products GmbH
SUSE
Maxfeldstr 5
90409 Nürnberg, Germany
e-mail: owen.synge@jaysnest.de

Abstract. The new federated Cloud infrastructures demand the development of specific utilities and technical solutions. One of the most crucial aspects is the virtual image management and distribution system. This paper summarizes the work carried out within the EGI FedCloud taskforce during the last year to deploy a sustainable, secure and scalable federated VM image management system.

Keywords: Cloud computing, federated Cloud, image management, virtual machines

Mathematics Subject Classification 2010: 68-68N01

1 INTRODUCTION

The number of projects based on Cloud IaaS resources are increasing each day. The European projects are making efforts to help abstract away the differences between different Cloud APIs and allow resource consumers to be independent of Cloud infrastructure. These Cloud agnostic frameworks will become the new Rosetta stone for Cloud developers and site administrators.

One of these european projects which is consolidating its Cloud services is EGI-InSPIRE [1] (Integrated Sustainable Pan-European Infrastructure for Researchers in Europe). EGI-InSPIRE is composed by a federation of national resource infrastructure providers which includes Grid and now Cloud resources. Federating these Cloud resources is a major priority for EGI that is coordinated by EGI Federated Cloud Taskforce [2].

Projects like EGI are using different APIs, Cloud frameworks, storage systems etc. However this technology agnosticism increases the interoperability challenges. One of the most pressing issues is the managing of VM images (VMIs) within a federated Cloud infrastructure. A federated Cloud increases the difficulty of this management, since it uses different Cloud APIs, image formats, contextualisation mechanisms and so on.

With Virtual Appliances (VAs) like VMIs on Cloud IaaS resources the complications due to the integration of multiple components and dependencies of services can be managed by the appliance creator or maintainer without need for the Cloud resource provider to be involved. Cloud IaaS has shown great commercial success, new Cloud software stacks are being developed, and many companies and research institutes are deploying these Clouds for both public and private use. The EGI-Inspire project is an excellent testbed which brings together many sites, running multiple IaaS Cloud implementations within a production Federated Cloud infrastructure.

The interoperability of Appliance distributions is one of the first issues in the process of creating a federated Cloud. This paper is focused on some of the concerns surrounding Virtual Appliance Image Management in a federated Cloud environment following the HEPiX Virtualisation Working Groups (HVWG) [17] recomendations.

A federated Cloud using a heterogeneous Cloud frameworks ecosystem (mixing OpenStack, OpenNebula, WNoDES [21], etc) has different APIs, Appliance formats, and contextualisation mechanism for each Cloud implementation. The tools used to evaluate the HVWG's [17] proposals did not support any specific Cloud infrastructures at the start of this process, so adapters had to be developed for each IaaS System supported.

The primary goal of an EGI-federated Cloud task force is to investigate the potential of IaaS Clouds as a revolutionary upgrade to Grid Computing's use ability. IaaS Clouds have been been deployed at many sites where they were actively used by communities not suited to Grid Computation [22].

The secondary goals include an unification of IaaS resources (from a user experience and from resource sharing) and extending the quality of service beyond the capabilities of a financially constrained resource provider. This is very similar to the

goals of Grid computing but making use of IaaS platforms.

The aim of the HVWG was to propose policies so that resource providers have a way to control and manage VAs like Virtual Machine Image's (VMIs) provided by experimenters that need to be executed in a trusted environment, similar to the current computing environment provided under Grid computing. The authors of this paper are members of the EGI-federated Cloud task force collaborating with a developer and a former member of the HVWG [17].

This paper is organised in the following sections. First, Section 2 presents the state of the art regarding Cloud framework VM management, Section 3 describes the VMcaster/VMcatcher image management tools, the new tools proposed by HVWG for image management. Section 4 explains how VMcatcher event handlers work, and how VMcatcher plugins support different Cloud frameworks like OpenNebula or OpenStack. Finally Section 5 will present our conclusions and planned future work.

2 RELATED WORK

The use of Cloud infrastructures it is a very promising field in science, but integrating Clouds with the Grid is a challenge, as related on Dillon et al. [3]. Goasguen et al. [13] present the results of an internal production Cloud service in CERN and suggest to expand it to other Grid sites. Zhao et al. [12] present an infrastructure of a dozen computing sites using OpenNebula as the management solution. It concludes that Clouds are very useful for science, but there are still many performance issues to be resolved. Hoffa et al. [4] reached similar conclusions regarding Cloud vs local deployments.

There are also many works that compare the different solutions for VM Management, like Xiaolong et al. [14], which compare OpenNebula and Openstack, and Laszewski et al. [8], which does a more complete survey including Eucalyptus, Nimbus and some other frameworks. The existence of numerous trade-offs and the fragmented market for these tools motivated us to support federated environments.

In the realm of security, our solution emphasizes the authentication of users and the validation of VMs. There are other works on the area, but some, like Xi et al. [6] are concerned more with running trusted VMs on on untrusted environments, which can be seen as the opposite problem, and many others, like Schwarzkopf et al. [11] are concerned with improving the internal security of VMs maintained by Cloud users instead of infrastructure operators.

There are still other comparable solutions, Lagar-Cavilla et al. [5] use a non-local fork mechanism to spawn many copies of a VM across many sites, but this method would be at odds with current Grid practices. Diaz et al. [9] have a similar system that bridges OpenNebula and OpenStack, but it uses the Amazon EC2 API, which has licensing issues preventing us for using it, and does not address the authorization and validation of VMs. On a more partial resemblance, Maurer et al. [10] also automate some aspects of VM management and update it using an autonomous system, and Django et al. [7] change the context of VMs on the fly to do load balancing and

improve brokering. This last functionality would be invaluable for Grid operators, which must frequently tend to processes that get stuck due to unrealistic brokering requirements, and would also avoid many downtimes due to reconfiguration.

3 APPLIANCE LIFECYCLE MANAGEMENT USING IMAGELISTS

The HVWG proposed a message system to decouple appliance maintainers from Cloud implementations. This decoupling could be used to hide the heterogeneous nature of a federated Cloud. The HVWG proposal is based upon SMIME and so is not dependent of the method of signing and is applicable to both PGP or X.509 signatures.

The imagelists are published via HTTP and polled by the subscribers. Then their signature is checked, subscribed and referenced. The VMIs are downloaded and their integrity checked using the secure hash. If the appliance VMI is valid the appliance is then able to be contextualised and then instantiated automaticaly if the resource provider supports this.

By using this mechanism an appliance VMI can be checked for validity, all imagelists are signed and they provide a version number and expiration date. If the imagelist does not satisfy these requirements the VMI is not downloaded and it is set as not valid.

The subscription process is similar to Debian's package managing tool (*aptitude*) or the rpm update management (*yum*) utilities or a podcast subscriber. Publishing is similar to *create-repo* which publishes rpm packages rather than virtual machines Appliances. The HVWG's proposals includes two main differences, each VMI is uniquely identified by an UUID, and those no longer present in an imagelist indicates its expiration/revocation rather than an imagelist error.

VMcaster [15] and VMcatcher [16] are two different tools to generate and subscribe to virtual machine imagelists created and designed by HVWG members. These tools use an internal database (in the same way that the internal databases used by podcast syndication or a Linux package manager) where VMIs information and lists are stored into an internal SQLite database (see figure 1). SQLite has proved more than adequate for the low transaction rate of a subscriber and so deployment issues are just backing up a database file.

These tools try to match the requirements set by the now ended HEPiX virtualisation working group [17]. During the last year the EGI federated Cloud task force has recommended VMcatcher installation in its Cloud resource providers.

In this case since the software is made with the Grid in mind and to avoid *man in the middle* security issues, VMcaster/VMcatcher tools support the X.509 certificate authentication model. All the imagelists are signed by an authenticated endorser with his/her personal X.509 certificate. This means all VMIs are referenced by a Virtual Machine Image List which contains a secure hash (SHA512) signed using X.509 personal certificates (provided by the endorser). These Virtual Machine Image Lists are published, and interested sites can subscribe to the lists in the resulting

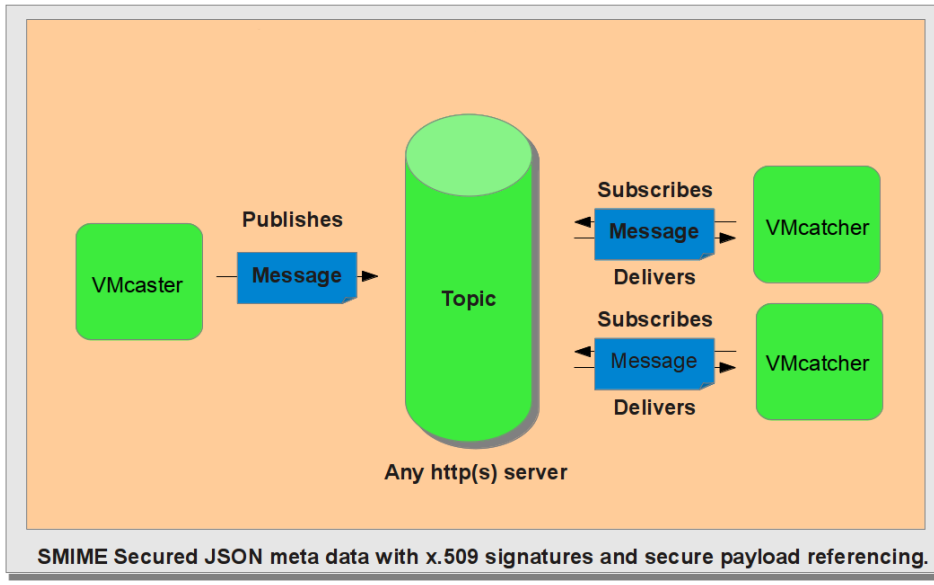

Figure 1. VMcaster and VMcatcher infrastructure.

catalogue.

Another important feature of VMcaster/VMcatcher is the support of Cloud framework agnostic tools, i.e. these tools do not depend on the Cloud solution used by the sites. Besides it can be integrated with different frameworks using different plugins to use the new VMIs directly from for example, OpenNebula or OpenStack (see section 4 for more information).

3.1 Publishing imagelists

Publishing VMIs according to the HVWG's proposals is as simple as generating the meta data and signing it with standard SMIME signature routines. No new software is needed for this task. Due to the amount of meta data recommended by the HVWG and the possibility that users may want to add their own additional information, it is expected that users employ tools for generating the necessary JSON file.

Imagelists use a JSON meta data file signed with SMIME with a secure hash. However the information recommended by the HVWG is burdensome to enter by hand.

VMcaster is a simple tool for publishing, managing and updating VMI imagelists which follow the HVWG specifications. It was released after the EGI federated Cloud taskforce started using the HVWG's recommendations.

All the metadata created by VMcaster are signed and trusted with a X.509 certificate. This provides a mechanism by which a VMI can be checked for validity by

any subscriber. Any user can check the imagelist expiration date, if it was revocated or has been tampered by a third party. All VMIs and imagelists have an Universal(ly) Unique Identifier (UUID). These UUID's should be globally unique and, consequently, the UUID should be generated using a UUID generator using suitable seeds. As example for Debian, Redhat and Scientific Linux users can execute the following UUID generator:

```
$ uuidgen

dfc470ab-0845-4c3b-bc6a-02f990388a17
```

We can use the new UUID with VMcaster to create a empty imagelist:

```
$ vmcaster --select-imagelist dfc470ab-0845-4c3b-bc6a-02f990388a17 \
--add-imagelist
```

This command generates an empty imagelist in our local *vmcaster.db* database (not published yet) with a few predefined objects. We can query the imagelist UUID to see the current object list. The database information is shown in JSON format:

```
$ vmcaster --select-imagelist dfc470ab-0845-4c3b-bc6a-02f990388a17 \
--show-imagelist
{
    "hv:imagelist": {
        "dc:identifier": "dfc470ab-0845-4c3b-bc6a-02f990388a17"
    }
}
```

The object *dc:identifier* contains always an UUID and it is included in VMIs or imagelists. At this moment the imagelist does not include any relevant information but thanks to the VMcaster utility the endorser can introduce new metadata and information. The most important objects are: the VMI title, its description, its endpoint (a valid *url*) and also the endorser unique certificate DN. Only a trusted endorser can modify or update an VMcaster imagelist to include or remove VMIs. These values can be included into the internal database running these commands, as example:

```
$ vmcaster --select-imagelist <image_list_UUID> \
--key-set-imagelist "dc:title"\
--key-value-imagelist "New image list"

$ vmcaster --select-imagelist <image_list_UUID> \
--key-set-imagelist "dc:source" --key-value-imagelist "CESGA"

$ vmcaster --select-imagelist <image_list_UUID> \
--key-set-imagelist "dc:description"\
--key-value-imagelist "My image list for internal users"
```

```
$ vmcaster --select-imagelist <image_list_UUID> \
--key-set-imagelist "hv:uri" --key-value-imagelist \
"http://Cloud.cesga.es/files/image.list"

$ vmcaster --select-endorser \
"/DC=es/DC=irisgrid/O=cesga/CN=alvarosimon" \
--key-set-endorser "dc:creator" --key-value-endorser "Alvaro Simon"
```

Endorsers can also import imagelists (in JSON or MIME format) to streamline the imagelist creation. At this moment the endorser can include new metadata into the imagelist catalogue. The VMI insertion procedure is similar to imagelist creation *vmcaster –select-image <UUID> –key-set-image <IMAGE TAG> –key-value-image <VALUE>*. The endorser only has to generate a new UUID and insert the relevant objects, in this case:

- *dc:title*: Image title name
- *sl:comments*: Includes VMI comments (user login and password, software included etc).
- *sl:osversion*: The Operating System version as LSB compliant. I.e. Scientific Linux release 6.4 (Carbon).
- *sl:arch*: System architecture (x86_64, i386 etc).
- *sl:os*: Operating System name (Ubuntu, Debian, RedHat..).
- *hv:uri*: VMI location endpoint. The VMI be accesible from this url. As example `http://Cloud.cesga.es/images/debian-6.0.5-x86_64-base.qcow2`
- *hv:format*: VM image format (QCOW2, RAW)

All the new VMIs should be assigned to a imagelist, we can include the same VMI into different imagelists. To add a new VMI to a specific imagelist:

```
$ vmcaster --select-imagelist <IMAGE_LIST_UUID> \
--imagelist-add-image --select-image <IMAGE_UUID>
```

VMcaster can be configured to syncronize and upload local VMIs to a specific server endpoint. This mechanism allows VMcaster users to upload VMIs and imagelists to a web server in an automated way. This information is located into a configuration file (*/etc/vmcaster/vmcaster.cfg*). This mechanism is very useful for endorsers as they can use this configuration file to include several servers to keep the imagelists up to date in a short period of time. *vmcaster.cfg* file uses this schema:

```
[SERVER NAME]
server = "myserver.org"
protocol = "scp"
uriMatch = "https://myserver.org/"
uriReplace = "user@myserver.org:/var/www/html/"
```

In this case VMcaster will use the `myserver.org` web page to upload any VMI or imagelist. *vmcaster.cfg* accepts different communication protocols such scp, GSIdCap [18] or a local transmission. If this configuration file is set, VMcaster will upload VMIs automatically using this command:

```
vmcaster --upload-image <local_image_path> \
--select-image <IMAGE_UUID>
```

VMcaster detects the VMI size and generates a SHA512 hash for each uploaded file. When this process is complete the updated information is included into the VMcaster database automatically. At the end the information can be gathered from the local database. The information can be displayed in JSON format, a future subscriber can identify the available VMIs, the imagelist creation/expiration dates or the endorser DN.

When the imagelist is ready and updated the endorser can publish it to all sites included into the *vmcaster.cfg* file. The procedure is quite similar to the VMI uploading, but in this case, the VMI must be signed by the endorser certificate to validate its authenticity. VMcaster asks for user certificate password and uploads the imagelist to its final endpoint, this can be done with a single command:

```
$ vmcaster --select-imagelist <IMAGE_LIST_UUID> --upload-imagelist
```

This procedure allows different VMI endorsers to distribute and update VMI catalogs on different endpoints and sites (which is suitable for a federated architecture). Besides, all imagelists have endorsed information about endorser certificate public key, the VMI download endpoint, initial global validity, etc. That means valid VMIs can be selected, downloaded and instantiated and tested by different resource providers.

The new images should be verified by the imagelist endorser but federated Cloud resource provides have the final say about its inclusion into their Cloud image repositories. VMI subscription task is done by VMcatcher utility described in the next section.

3.2 Subscribing to image lists

VMcatcher [16] is currently the only HVWG compliant imagelist subscriber, and only supports JSON meta data and X.509 signatures. Since the Grid community already use the International Grid Trust Federation [23] supporting just X.509 is acceptable for the current stage for the Federated Cloud task force so these limitations have not hindered this investigation. It is recommended that future HVWG imagelist providers support PGP and X.509 signatures.

VMcatcher allows subscription of VMIs by UUID so that a resource provider can provide contextualised appliance VMIs from the most recent appliance VMI. Resource providers can configure their own imagelists and select VMIs from different imagelists or a Marketplaces, these are then validated in compliance with the HVWG

security policies and cached locally. VMcatcher does not support any Cloud explicitly so further work is needed in this area (see next section).

VMcaster can be configured to launch applications on VMI status change events such as expiry or new images becoming available. The federated Cloud task force has developed such integration handlers for managing Appliance Life cycle management with OpenStack and OpenNebula.

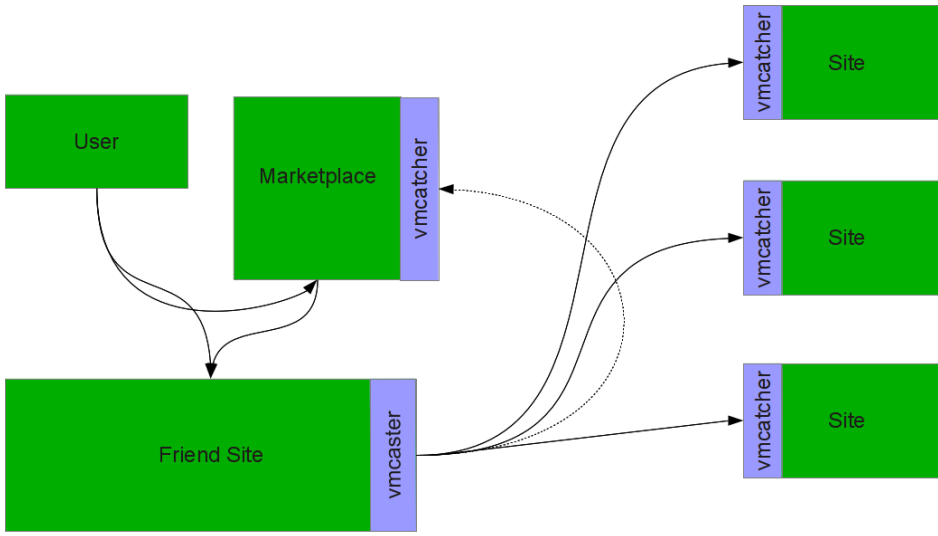

Figure 2. EGI FedCloud Image Management model

Using this utility users can select and download trusted VMIs (see figure 2). This utility caches the selected VMIs in a imagelist, validates the list with X.509 based public key cryptography, and also checks the VMIs SHA512 hashes.

VMcatcher and VMcaster work in a similar way. They are based upon a local database that stores subscriptions to VM imagelists, registered endorsers, and which images belong to which subscriptions. This enables VMIs to be selected for subscription. An user can select the desired VMI from the imagelist and then set the desired VMI subscriptions. Subscribed VMIs can be downloaded, verified and cached. VMcatcher also verifies if the cached images have expired or not. If an VMI is invalid or it has expired it is moved to an expiry directory. The image consumer must trust a endorser, in this case users can include and confirm their trust in a VMI endorser based on his/her X.509 certificate Distinguished Name (DN). To include a new trusted endorser:

```
$ vmcatcher_endorser --create --endorser_uuid='Alvaro Simon' \
--subject='/DC=es/DC=irisgrid/O=cesga/CN=alvarosimon' \
--issuer='/DC=es/DC=irisgrid/CN=IRISGridCA'
```

And now the user only has to download the desired imagelist from the endpoint and import it into a local VMcaster database:

```
$ wget http://Cloud.cesga.es/files/image.list
$ vmcatcher_subscribe -s file:////‘pwd‘/image.list
```

At this point the user can show and select any VMI UUID from the new imagelist to be downloaded. The imagelist may vary (VMI endorse includes new images, revoke old ones, etc), in any case the local imagelist database should be updated frequently executing the command *vmcatcher_subscribe -U*. This command checks the imagelists endoint and updates the local database accordingly. After image selection the VMIs can be downloaded and updated to a local cache running:

```
$ vmcatcher_cache
```

The new downloaded VMIs are stored into *<VMCATCHER_DIR>/cache* directory, meanwhile revoked or expired images are moved to *<VMCATCHER_DIR>-/cache/expired* directory. This mechanism prevents that old or revoked VMIs are used by mistake.

4 VMI MANAGEMENT EVENT HANDLERS

VMcatcher and VMcaster are useful tools to disseminate and keep updated our VMIs but they do not interact with Cloud frameworks directly. VMcatcher was written in Python and generates pre-defined events that can be received by an asynchronous callback subroutine or event handler. These pre-defined events can be used by the Cloud frameworks to perform different actions.

Fortunately the Cloud community has developed event handlers to interact with the most popular frameworks like OpenNebula or CloudStack. OpenNebula event handler [19] was developed by the CESGA team and currently is available from the VMcaster repository. The *vmcatcher_eventHndlExpl_ON* package provides a new python and cron script which detects VMcatcher events. This script detects several VMcatcher event types. In this case the OpenNebula handler only waits for a new Expire or Available VMcatcher event. If VMcatcher raises a *AvailablePostfix* event, this is detected by *vmcatcher_eventHndlExpl_ON* event handler and reads the new VMI attributes such UUID, image name, description and image format.

If a new VMI is downloaded (*AvailablePostfix* event), the OpenNebula event handler gets the VMI information, generates a new OpenNebula template and includes the new image into the local OpenNebula datastore (see figure 3). For security reasons, the new VMIs are not public, they are only available for the oneadmin user. The OpenNebula administrator should verify the new VMI first (checking its contextualisation script, if the image is executed correctly, etc).

After this period of time the VMI status is changed to be available for external users. Moreover, if VMcatcher detects an image revocation the OpenNebula event handler searches the VMI UUID from OpenNebula image database and this UUID

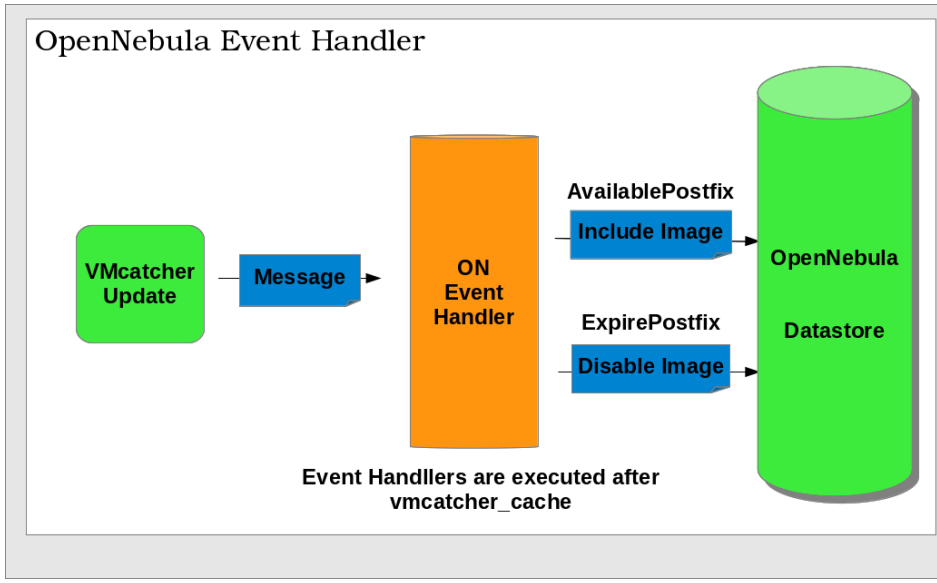

Figure 3. OpenNebula event handler.

is set to disable status. The image is not removed by the event handler, it should be removed by the site administrator from the OpenNebula datastore.

The OpenNebula event handler is not the only one available. OpenStack administrators can also use the Glancepush [20] service to keep their local VMI catalog updated. This service was developed at IN2P3 and it works in a similar way than the OpenNebula event handler. In this case Glancepush updates the OpenStack Image Service (Glance) if it detects any VMI change from VMcatcher tool. The new package (*glancepush-vmcatcher*) is available from the IN2P3 ftp server, and it only requires a working glance service and an OpenStack user account to push images into the catalog.

5 CONCLUSIONS AND FUTURE WORK

The management of VMIs is a critical task within a federated Cloud architecture. It involves certain saecurity standards and special scalability and availability and stability requirements. Taking into account these needs the EGI FedCloud taks force has chosen the VMcaster and VMcatcher utilities to distribute and validate VMIs between the resource providers. The FedCloud resource providers are using heterogenous Cloud frameworks ecosystems (OpenStack, OpenNebula, WNoDES [21], etc). This kind of federated infrastructure requires agnostic tools. Fortunately as we have explained in this paper, VMcatcher can be used by any Cloud framework to distribute and update VMIs in a transparent way. The new VMI management tools

are being successfully used by EGI FedCloud providers since last year and it will be used in more use cases in the near future.

The greatest benefit for the EGI-federated Cloud task force is to simplify the management of IaaS systems with respect to the native IaaS Clouds own APIs. However this Cloud implementation neutrality also increases the interoperability challenges. The HVWG proposal is esentially federated with a distributed system, has no single point of failure or any single service in control, and complies with a policy that was agreed by the mebers of the HVWG as a secure best practice. The use of a simple signed lists of endorsed Appliances in these proposals is equally applicable to all PKI infrastructures supported by SMIME.

We believe that the HVWG proposal to manage Appliance Lifecycle, with signed messages is valid and expressive enough for federating appliances. Appliance VMI distribution and management is typically I/O bound and so asynchronous in nature signed aplliance imageslist provide a simple and clear way of managing appliance images that can be audited by resource providers. We found the clear definition of valid and invalid/expired VMIs, and the distributed nature of Appliance lists a natural fit to a federated Cloud and worth testing.

By testing the use of HVWG proposed VMI distribution system, with the tools VMcaster and VMcatcher, and developing event handlers for VMcatcher, the Federated Cloud Task force has proved that this approach is a distributed system for VMI distribution, for both OpenStack and OpenNebula resource providers. The FedCloud task force has validated VMcaster and VMcatcher utilities as production grade implementation, and will continue to use them to distribute and validate VMIs.

The EGI FedCloud task force is committed to support these new image managements tools in the future and development work is still ongoing within the project. The GRNET partner is developing a new Virtual Appliance Marketplace based on EGI Applications Database (EGI AppDB), a project service which stores and provides to the public information about Grid software solutions created by the scientists and programmers. This service was extended to create a Virtual Appliance Marketplace to bring a new category of software entries or VAs, which are VMIs designed to run on a virtualized platform. AppDB's Virtual Appliance Marketplace was integrated with the existing HVWG VMCaster/VMCatcher framework.

EGI Marketplace uses VMcatcher to download and update the new VMIs included into an EGI imagelist and provides this list to FedCloud users. The basic image management features were embedded in the Marketplace, such as creating, publishing, enabling/disabling, and archiving or deleting VAs or VMI versions. Besides, a friendly site can include its own imagelist to be published by the Marketplace. With the new service any authorised user (using her/his X.509 certificate) will be able to submit or upload VA version metadata (in the form of signed imagelists).

Project RPs can subscribe to specific VAs or to the Marketplace imagelist. Using this mechanism a trusted FedCloud user group, Virtual Organization or experiment can disseminate its VAs or VMIs among project RPs in a short period of time. FedCloud Resource Providers site admins only have to subscribe to the new VMIs using VMcatcher, and the VMIs will be available automatically thanks to the

VMcatcher event handlers.

This management model is still under development and testing, but it will be available in production and usable by EGI users in the next months.

6 ACKNOWLEDGEMENTS

This work is partially funded by the EGI-InSPIRE (European Grid Initiative: Integrated Sustainable Pan-European Infrastructure for Researchers in Europe) is a project co-funded by the European Commission (contract number INFSO-RI-261323) as an Integrated Infrastructure Initiative within the 7th Framework Programme. EGI-InSPIRE began in May 2010 and will run for 4 years. Full information is available at: `http://www.egi.eu/`.

References

[1] The EGI-InSPIRE project.
`https://www.egi.eu/about/egi-inspire/`, 2013
Last visit on April 2014

[2] EGI Federated Clouds Task Force.
`https://wiki.egi.eu/wiki/FedCloud-tf:FederatedCloudsTaskForce`, 2013
Last visit on April 2014

[3] Dillon, T. et al.: Cloud Computing: Issues and Challenges. Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference, pp. 27–33

[4] Hoffa, C. et al.: On the Use of Cloud Computing for Scientific Workflows. eScience, 2008. eScience '08. IEEE Fourth International Conference, pp. 640–645

[5] Lagar-Cavilla: SnowFlock: rapid virtual machine cloning for Cloud computing. Proceedings of the 4th ACM European conference on Computer systems, EuroSys 2009, pp. 1–12

[6] Chunxiao, L. et al.: A Trusted Virtual Machine in an Untrusted Management Environment. Services Computing, IEEE Transactions, Vol. 5, 2012, No. 4, pp. 472–483

[7] Armstrong, D. et al.: Runtime Virtual Machine Recontextualization for Clouds. Lecture Notes in Computer Science, Euro-Par 2012: Parallel Processing Workshops, Vol. 7640, pp. 567–576

[8] Laszewski, V. et al.: Comparison of Multiple Cloud Frameworks Cloud Computing (CLOUD), 2012 IEEE 5th International Conference, pp. 734–741

[9] Diaz, J. et al.: Abstract Image Management and Universal Image Registration for Cloud and HPC Infrastructures, Cloud Computing (CLOUD), 2012 IEEE 5th International Conference, pp. 463–470

[10] Maurer, M. et al.: Adaptive resource configuration for Cloud infrastructure management. Future Generation Computer Systems, Vol. 29, 2013, No. 2, pp. 472–487

[11] Schwarzkopf, R. et al.: Increasing virtual machine security in Cloud environments. Journal of Cloud Computing, Vol. 1, 2012, No. 1, pp. 1-12

[12] Yong, Z. et al.: Designing and Deploying a Scientific Computing Cloud Platform. Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference, pp. 104–113

[13] Goasguen, S. et al.: LxCloud : a prototype for an internal Cloud in HEP. Experiences and lessons learned. Journal of Physics: Conference Series, Vol. 396, 2012, No. 3

[14] Wen, X. et al.: Comparison of open-source Cloud management platforms: OpenStack and OpenNebula. Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference, pp. 2457–2461

[15] VMcaster VM image publication tool.
`https://github.com/hepix-virtualisation/vmcaster`, 2013
Last visit on November 2013

[16] VMcatcher image subscription tool.
`https://github.com/hepix-virtualisation/vmcatcher`, 2013
Last visit on November 2013

[17] Tony Cass: The HEPiX Virtualisation Working Group: Towards a Grid of Clouds. Journal of Physics: Conference Series, Vol. 396, 2012, No. 3

[18] Mathias De Riese: The dCache Book.
`http://www.dcache.org/manuals/Book/`, 2013
Last visit on November 2013

[19] OpenNebula event handler.
`https://github.com/grid-admin/vmcatcher_eventHndlExpl_ON`, 2013
Last visit on November 2013

[20] OpenStack Glancepush service.
`https://github.com/EGI-FCTF/glancepush/wiki`, 2013
Last visit on November 2013

[21] Salomoni, D. et al.: WNoDeS, a tool for integrated Grid and Cloud access and computing farm virtualization. Journal of Physics: Conference Series, Vol. 331, 2011

[22] Zhang, S. et al.: The comparison between Cloud computing and grid computing. Computer Application and System Modeling (ICCASM), Vol. 11, 2010, pp. 72–75.

[23] The International Grid Trust Federation.
`http://www.igtf.net/`
Last visit on November 2013

[24] EGI Applications Database.
`http://appdb.egi.eu/`
Last visit on November 2013

Álvaro Simón is a Grid and Cloud senior technician at CESGA in Santiago de Compostela (Spain) since 2007. He gradutated in physics in 2005 at the Universidad de Santiago de Compostela (USC). He has participated in several international projects in the distributed computing area including int.eu.grid and EGEE (II and III) projects. Currently he is the task leader of EGI TSA2.3 software verification process and EGI FedCloud brokering workgroup.

Owen Synge is an experienced software developer. He graduated in biochemistry in 1996 at the University of Bristol and in Compute Science in 1997 at the University of York. He has led the design, implementation, and the deployment of many software products in different sites from Universities to International research centers, for the Large Hadron Collider Computing Grid, which is now successfully investigating the Higgs Boson. He has worked as a professional developer on various Linux platforms for 11 years, for large scale research computing problems. Currently he is working at SUSE as expert developer in distributed storage.